PL-TR-97-1139 PL-TR-97-1139

PARALLELIZATION OF TWO- AND THREE-DIMENSIONAL FAST FOURIER TRANSFORMS

Paul Bennett

University of New Mexico
1601 Central Avenue
Albuquerque, New Mexico 87131

February 2001

Final Report

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

AIR FORCE RESEARCH LABORATORY
Directed Energy Directorate
3550 Aberdeen Ave SE
AIR FORCE MATERIEL COMMAND
KIRTLAND AIR FORCE BASE, NM 87117-5776

PL-TR-97-1139

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data, does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report has been reviewed by the Public Affairs Office and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nationals.

If you change your address, wish to be removed from this mailing list, or your organization no longer employs the addressee, please notify AFRL/DEBI, 3550 Aberdeen Ave SE, Kirtland AFB, NM 87117-5776.

Do not return copies of this report unless contractual obligations or notice on a specific document requires its return.

This report has been approved for publication.

Valerie B... for

BRIAN BEVERIDGE, Capt USAF
Project Manager

FOR THE COMMANDER

R. Earl Good

R. EARL GOOD, SES
Director, Directed Energy

CHRISTOPHER S. WASHER, Lt Col, USAF
Chief, Advanced Optics and Imaging Division

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
27-02-2001	Final Report	May-Sep 1997

4. TITLE AND SUBTITLE
Parallelization of Two- and Three-Dimensional Fast Fourier Transforms

5a. CONTRACT NUMBER F29601-96-C-0006

5b. GRANT NUMBER N/A

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Paul Bennett

5d. PROJECT NUMBER ARPA

5e. TASK NUMBER NA

5f. WORK UNIT NUMBER AA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The University of New Mexico
1601 Central Avenue
Albuquerque, New Mexico 87131

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AFRL/DEBI
550 Lipoa Parkway
Kihei, Maui, HI 96753

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S) PL-TR-97-1139

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Large computational programs employing multi-dimensional spectral techniques for numerical modeling or image processing rely heavily on the use of Fast Fourier Transforms (FFTs). This report describes work performed to port Numeric Algorithms Group, Ltd. (NAG) two-dimensional and three-dimensional FFTs to the IBM SP-1 distributed parallel computing platform to transform arrays of almost any dimension upon many processors, while achieving significant speed-up factors. A description of the parallelization strategies used, observed performance, and conclusions derived is provided. The parallel FFT performance achieved is significant in that it allows for flexible, portable, and robust use of spectral and image processing codes across arbitrary arrays of processors. This further allows for more basic scientific research and production computing with greater flexibility than before. The work described by this report indicates that there is a great potential in implementing flexible and efficient FFTs that can meet the ever-increasing demands of future image and digital signal processing systems.

15. SUBJECT TERMS
parallel, parallelization, Fast Fourier Transforms, FFTs, scientific computing, image processing, digital signal processing

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
Unclassified	Unclassified	Unclassified	UL	8	Capt Brian Beveridge
					19b. TELEPHONE NUMBER *(Include area code)* (808) 891-1702

1 Introduction

The Fast Fourier Transform (FFT) plays a central role in scientific computing. For example, it is used in image processing, digital signal processing, and the numerical solution of differential equations using spectral and pseudo-spectral methods. Many vendors, such as IBM and NAG, have implemented FFTs in their proprietary libraries. However, these implementations are far from complete. IBM's Parallel Engineering and Scientific Subroutine Library is written for the Power RS6000 chips. It does not allow every sequence length to be transformed, and the sequence lengths that may be transformed must be divisible evenly by the number of processors used to perform the transformation [1]. But the NAG FFTs have many more allowable lengths [2]. However, these FFTs are implemented for sequential and vector computers, not for distributed memory computers. This project ports the NAG 2-d and 3-d FFTs, C06FUF and C06FXF, respectively, to the IBM SP-1, to transform 2-d and 3-d arrays of almost any dimension's upon arbitrarily many processors. Section 2 presents the parallelization strategies used, section 3 reports performance, section 4 presents a discussion of the performance, and section 5 is a summary and conclusion.

2 Parallelization

The NAG routines are implementations of the self-sorting Stockham algorithm applied to multiple transforms of mixed-radix lengths [2],[3]. The various radices used are 2, 3, 4, 5, 6, and odd numbers not divisible by 2, 3, 4, 5, or 6. The FFT transformation occurs along the last dimension of the data in storage, so that the arrays are transformed one dimension at a time, then transposed and so on until the entire array has been transformed. Use has been made of a branching statement in each radix subroutine which minimizes the number of short do-loops executed in the course of the computation, at the cost of increased striding through the data.

The FFT algorithm necessitates that the data along the dimension to be transformed be located all on one processor at the time of transformation. We require a reasonably good balance of work as well, so that the arrays must be distributed as evenly as possible. However, in general, the array dimensions won't be divisible evenly by the number of processes taking part in the transformation. This means that at least one processor will wait for a while until all of the other processors are finished with their share of the computation before any communication can take place. Therefore, for a matrix of dimension M x N, processors 1, 2, ..., P-1, are assigned M/P rows, and the last processor, P, gets the remaining rows. For a three-dimensional array of size M x N x O, the first P-1 processors get M/P slices, each of size N x O, and the last processor gets the remaining slices. This method of data is referred to as *row-block*.

The algorithms for the two FFTs then proceed as follows. For the 2-d FFT, each processor transforms its data by performing M/P transforms of length N. The

global array is then block-transposed using MPI_ALLTOALLV. After a local transpose and reordering, the data is again transformed using N/P FFTs of length M . The array is then globally block-transposed again, using MPI_ALLTOALLV, and the data locally transposed and reordered, concluding the parallel 2-d FFT. The FFTs used at each stage are the above multiple self-sorting Stockham routines of mixed radix.

The 3-d FFT is similar, with N x M/P FFTs of length O being applied first. Then a local transpose is performed, followed by O x M/P FFTs of length N. MPI_ALLTOALLV is used to perform a global block-transposition. Then a local transpose and reordering of the data is followed by O x N/P FFTs of length M. We conclude the FFT by invoking MPI_ALLTOALLV and following that with a final local transpose and reordering of the data.

Double precision complex arrays are used for the sending and receiving buffers, in order to avoid the extra latency otherwise required.

2.1 Analysis

C06FUF has a complexity of roughly $\Theta(MN \log(MN))$, while C06FXF has a complexity of roughly $\Theta(MNO \log(MNO))$ [2]. To perform these transforms in parallel will require 2 communication phases, but will divide the amount of computation by P, the number of participating processes. During each communication phase, each processor will send at most 2(M/P)(N/P) real numbers to each of the other P-1 processors, and therefore receive that many real numbers from each other process. This will be modeled by the processor array performing P simultaneous scatters of N x M/P complex numbers, followed after some computation by another P simultaneous scatters of M x N/P complex numbers. We are assuming here that the communication network is capable of this. This gives a complexity of about $\Theta((MN/P)\log(MN) + \log(MN/P))= \Theta((MN/P)\log(MN))$ for PC06FUF. For PC06FXF, we get a complexity of $\Theta((MNO/P)\log(MNO))$.

Using latency β and bandwidth λ, the communication time for PC06FUF is estimated to be $2\beta \log P + \lambda(\log(NM/P) + \log(MN/P)$, or roughly $2\beta \log P + 2 \lambda \log(MN/P)$. Therefore PC06FUF is estimated to require $2\beta \log P + 2 \lambda \log(MN/P) + \chi (MN/P)\log(MN)$ time to compute, where χ is the operation time. In a similar way, PC06FXF should take roughly $2\beta \log(P) + 2 \lambda \log(MNO/P) + \chi (MNO/P)\log(MNO)$ time.

3 Results

The parallelizations of the routines C06FUF and C06FXF, namely PC06FUF and PC06FXF, were timed on the IBM SP-1 supercomputer at the Albuquerque Resource Center in early May, 1997. The individual nodes are RS6000 Power chips, connected via TB-2 adapters and a highspeed Omega network [4]. The subroutines are written in Fortran 77 and were compiled using IBM's Fortran compiler xlf 4.1 with optimization set at level 03. IBM's MPI library was used to supply the communication subroutines and timer. To time PC06FUF, multiple transforms of 2-d arrays of dimension M x M , where M = 100, 200, ..., 1000, were performed on P = 1, 2, ..., 8, processors. For PC06FXF, multiple FFTs of size M x M x M, M = 10, 20, ..., 100, were timed on P = 1, 2, ..., 8, processors. The number of times the routines were executed is shown in the following table.

Routine	10	20	30	40	50	60	70	80	90	100
PC06FUF	2000	1000	1000	100	50	50	50	50	20	20
PC06FXF	100	100	30	25	25	25	20	20	20	20

Table 1. Number of timings averaged per routine vs size of FFT. PC06FUF dimensions are 10 times the numbers in the top row. PC06FXF dimensions are the numbers in the top row.

Using [5], the latency beta is estimated to be 40 sec. for IBM's MPI_ALLTOALLV, the bandwidth of the TB-2 adapter and high speed switch is taken to be 40 Mbits/sec., and the computational speed of the Power chip 67.2 MHz.

These parameters were used to form Figures 1. and 2. in Appendix 1, which estimate the performance of PC06FUF and PC06FXF. Note that both plots show that for small problems, the performance does not scale well with the number of processors used, while for larger problems, we may expect nearly linear speedup.

The observed timings and the speed-up factors are presented in the following four tables, followed by plots of the observed speedup factors.

Dimensions	1	2	3	4	5	6	7	8	Line-Type
100 x 100	0.0307	0.0266	0.0211	0.0166	0.0156	0.0139	0.0116	0.0106	Solid
200 x 200	0.1685	0.1057	0.0769	0.0625	0.0737	0.0650	0.0574	0.0498	Dashed
300 x 300	0.5540	0.2638	0.1723	0.1308	0.1158	0.1014	0.1206	0.1137	Dot Dashed
400 x 400	1.46	0.5358	0.3642	0.2637	0.3776	0.2499	0.2185	0.2006	Dotted
500 x 500	1.82	0.8318	0.5233	0.3688	0.4826	0.4017	0.3553	0.3365	Solid w/Circles
600 x 600	2.91	1.23	0.8067	0.5751	0.4595	0.3743	0.4961	0.4306	Dashed w/Circles
700 X 700	4.92	2.16	3.11	1.06	1.25	1.33	0.7984	0.7151	Dot Dashed w/Circles
800 x 800	8.11	3.81	1.70	1.61	2.06	1.23	1.06	0.8836	Dotted w/Circles
900 x 900	8.11	2.74	1.89	1.34	1.13	0.9041	1.16	1.02	Dashed w/Plusses
1000 x 1000	11.0	3.97	2.59	1.91	2.42	1.89	1.56	1.33	Solid w/Plusses

Table 2. CPU timings for PC06FUF vs Number of Processors on the ARC's IBM Sp-1 with TB-2 Adapters. All entries are in seconds.

Dimensions	1	2	3	4	5	6	7	8
100x 100	1.00	1.15	1.46	1.85	1.97*	2.21*	2.65*	2.90*
200 x 200	1.00	1.59	2.19	2.70	2.29*	2.59*	2.94*	3.38*
300 x 300	1.00	2.10	3.22	4.24	4.78	5.46	4.59*	4.87*
400 x 400	1.00	2.72	4.01	5.54	3.87*	5.84*	6.68*	7.28*
500 x 500	1.00	2.19	3.48	4.94	3.77*	4.53*	5.12*	5.41*
600 x 600	1.00	2.37	3.61	5.06	6.33	7.77	5.87*	6.76*
700 x 700	1.00	2.28	1.58	4.64	3.94*	3.70*	6.16*	6.88*
800 x 800	1.00	2.13	4.77	5.04	3.94*	6.59*	7.65*	9.18*
900 x 900	1.00	2.96	4.29	6.05	7.18	8.97	6.99*	7.95*
1000 X 1000	1.00	2.77	4.25	5.76	4.55*	5.82*	7.05*	8.27*

Table 3. Observed speedups for PC06FUF vs Number of Processors on the ARC's IBM Sp-1 with TB-2 Adaptors.

Dimensions	1	2	3	4	5	6	7	8	Line Type
10 x 10 x 10	0.0022	0.0031	0.0026	0.0024	0.0020	0.0023	-----	-----	Solid
20 x 20 x 20	0.0462	0.0266	0.0177	0.0138	0.0116	0.0119	0.0091	-----	Dashed
30 x 30 x 30	0.1594	0.0739	0.0555	0.0650	0.0366	0.0313	0.0314	0.0274	Dot-Dashed
40 x 40 x 40	0.6899	0.3224	0.1471	0.1097	0.1473	0.0833	0.0739	0.0635	Dotted
50 x 50 x 50	0.9043	0.4303	0.3067	0.2460	0.1832	0.1644	0.1509	0.1528	Solid
60 x 60 x 60	1.57	0.8324	0.5760	0.4022	0.3260	0.2676	0.2645	0.2330	Dashed w/Circles
70 x 70 x 70	3.46	1.71	1.37	1.06	0.6989	0.6725	0.5164	0.4774	Dot Dashed w/Circles
80 x 80 x 80	10.02	4.02	1.88	1.70	2.24	1.16	1.01	0.8344	Dotted w/Circles
90 x 90 x 90	6.61	3.48	2.27	1.78	1.38	1.01	1.06	0.8868	Dashed w/Plusses
100 x 100 x 100	10.71	4.47	3.50	2.49	2.03	1.60	1.38	1.19	Solid w/Plusses

Table 4. CPU timings for PC06FXF vs Number of Processors on the ARC's IBM Sp-1 with TB-2 Adapters. All entries are in seconds.

Dimensions	1	2	3	4	5	6	7	8
10 x 10 x 10	1.00	0.71	0.85	0.92	1.10	0.96*	-----	-----
20 x 20 x 20	1.00	1.74	2.61	3.35	3.98	3.88*	5.08	-----
30 x 30 x 30	1.00	2.16	2.87	2.45	4.36	5.09	5.08*	5.82
40 x 40 x 40	1.00	2.14	4.69	6.28	4.68	8.28	9.34	10.86
50 x 50 x 50	1.00	2.10	2.95	3.68	4.94	5.50	5.99	5.92
60 x 60 x 60	1.00	1.89	2.73	3.90	4.82	5.87	5.94	6.74
70 x 70 x 70	1.00	2.02	2.53	3.26	4.95	5.15	6.70	7.25
80 x 80 x 80	1.00	2.49	5.32	5.89	4.47	8.64	9.92	12.01
90 x 90 x 90	1.00	1.90	2.91	3.71	4.79	6.55	6.24	7.45
100 x 100 x 100	1.00	2.40	3.06	4.30	5.28	6.69	7.76	9.00

Table 5. Observed speedups for PC06FXF vs Number of Processors at the ARC's IBM Sp-1 with TB-2 Adapters.

4 Performance Evaluation

The subroutines PC06FUF and PC06FXF perform better than expected, though there are two distinct sources of problems. For the first source, a careful examination of Tables 3 and 5 shows that there appears to be superlinear speedup in numerous entries. This can be attributed to the fact that the serial FFTs, C06FUF and C06FXF, are written for vector processing. One consequence of this implementation is that the performance on cache based processors is not likely to be optimal with respect to the data flow through cache. The reduced problem size then has the effect that the data flow through cache for the

distributed computation is much better for the sizes tested, giving the appearance of superlinear speedup.

For the second source, the speedup graphs, Figures 3 and 4 in Appendix A, show dips at apparently random locations. However, in light of the branching statements minimizing the number of short do-loops executed, some of this behavior can be understood. Briefly, all FFTs for 1 up through 4 processors will use the loop combination which turns out to load the data in from cache along the first index. This is the order that Fortran arrays are stored in memory, so the loads are most efficient and the do-loops take less time. As the number of processors increases, the branching instructions minimizing the number of short do-loops executed forces the computer to load the data along the second index, giving loads in non-unit strides, degrading the performance. The asterisks in Tables 3 and 5 above indicate those instances.

The difficulties with the degradation of performance due to the selection of do-loop orders can be corrected with more careful branching conditions. These can be included in the implementation of a different serial algorithm as the basic building block of the parallel NAG FFT. Specifically, according to Norm Troullier of IBM, the algorithm that NAG uses, decimation in frequency, or DIF, is not the best for cache-based processing. The decimation-in-time, or DIT, algorithms perform better on cache-based processors. This is due to the fact that the Power chip is equipped with multiply-add floating point units for which iterated data-independent multiply-add instructions can be pipelined, increasing the performance of code written with this capability in mind. The DIT algorithms can be implemented using repeated data-independent multiply-add instructions much more easily than DIF algorithms.

Improvements can also be made in the method of data distribution, making it possible to perform the FFTs in parallel in the event that several processors have fewer than M/P rows of length N, for PC06FUF, or M/P slices of dimension N x O, for PC06FXF. For example, the current implementation of PC06FXF makes it impossible to perform a 10 x 10 x 10 FFT using PC06FXF on 7 or more processors. The horizontal lines as entries in the timing and speedup tables indicate such instances. The data distribution would distribute the sequences to be transformed more evenly, giving a maximal difference of one in the number of rows or slices on each processor, for PC06FUF or PC06FXF respectively.

Finally, according to [6], there are several different types of Omega network. A better parallelization algorithm can be designed by using the specific network that the SP-1 uses to communicate. This should give more cost optimal data distribution and communication, further improving the parallelization.

5 Conclusion

The Fast Fourier Transform can be and is used in many calculations requiring convolution of data and functions, numerical solution of differential equations, and in image processing and digital signal processing. With the increase in computational capability and in the size and difficulty of problems needing solution comes a correspondingly greater need for more advanced and practical tools, including parallel FFTs. Many companies are seeking to fill these needs, but have not yet managed to satisfy them in the best way possible. This project continues a program of study and programming which seeks to fill the need for efficient FFTs of arbitrary length across arrays of arbitrarily many processors. This project should be counted as an initial effort, as it indicates that there is great potential in implementing an efficient and flexible parallel FFT.

6 Acknowledgments

I thank Marc Ingber of the Department of Mechanical Engineering and Paul Alsing of the Department of Physics for inviting me to take part in the DARPA-ISV project of which this project is a part, and Norm Troullier of IBM and the University of Minnesota for his remarks and guidance in porting and parallelizing C06FUF and C06FXF. I also thank Richard Luczak of NAG for his support and source codes, and my teachers David Greenberg and Bruce Hendricks of Sandia National Laboratory, Brian Smith of the High Performance Computing and Education Research Center, and Tom Caudell of the Department of Electrical and Computer Engineering for the ideas presented and used in this project.

References

[1] IBM, Parallel Engineering and Scientific Subroutine Library Guide and Reference, Release 2, IBM 1995, 1996.

[2] NAG, NAG Fortran Library Manual, Mark 17, Vol. 1, Numerical Algorithms Group Limited, 995.

[3] C. Van Loan, Computational Frameworks for the Fast Fourier Transform, Society for Industrial and Applied Mathematics, 1992.

[4] L.D. Fosdick, E.R. Jessup, C.J.C. Schauble, and G. Domik, An Introduction to High Performance Scientific Computing, MIT Press, 1996.

[5] V. Georgitsis, Message Passing Performance on SP Systems, Master's Thesis, UNM/ARC, 1996.

[6] V. Kumar, A. Grama, A. Gupta, and G. Karypis, Introduction to Parallel Computing, Design and Analysis of Algorithms, The Benjamin/Cummings Publishing Company, Inc., 1994.

DISTRIBUTION LIST

DTIC/OCP 8725 John J. Kingman Rd, Suite 0944 Ft Belvoir, VA 22060-6218	1 cy
AFSAA/SAMI 1570 Air Force Pentagon Washington, DC 20330-1570	1 cy
AFRL/VSIL Kirtland AFB, NM 87117-5776	2 cys
AFRL/VSIH Kirtland AFB, NM 87117-5776	1 cy
The University of New Mexico 1601 Central Avenue Albuquerque, New Mexico 87131	1 cy
Official Record Copy AFRL/DEBI/Capt Brian Beveridge	2 cys